单隐层神经网络输入权值的一种新算法 *

刘金澎，田大钢

(上海理工大学 管理学院，上海 200093)

摘　要：针对传统极端学习机输入权值与隐层阈值随机设定的问题，提出了输出值反向分配算法。算法在传统极端学习机的基础上，通过优化方法得到最优输出值分配系数，并利用最小二乘法确定网络输入参数。将本文算法应用到常用数据集进行实验，并与其他极端学习机改进算法进行比较，显示本文算法有良好的学习以及泛化能力，能够得到简单的网络结构，证明了算法的有效性。

关键词：极端学习机；单隐层神经网络；优化方法；输出值反向分配

中图分类号：TP183　　doi: 10.3969/j.issn.1001-3695.2018.07.0418

New algorithm for input weight of single hidden layer neural network

Liu Jinpeng, Tian Dagang

(*Business School*, *University of Shanghai for Science & Technology*, *Shanghai* 200093, *China*)

Abstract: This paper proposed a back distribution algorithm for output values, which was to solve the random setting problem about input weights and hidden layer thresholds on traditional extreme learning machine. On the basis of traditional extreme learning machine, the algorithm obtains the optimal output value distribution coefficient through the optimization method, and uses the least squares method to determine the network input parameters. Experiments on common data sets show that, the proposed algorithm not only achieves good learning and generalization ability compared with other improved extreme learning machine algorithms, but also can obtain a simple network structure and it is effective.

Key words: extreme learning machine; single-hidden layer neural networks; optimization; output back distribution

0　引言

极端学习机（extreme learning machine，ELM）由 Huang 等人 2006 年首次提出，是一种新型的单隐层前馈神经网络模型，其运算速度是传统前馈神经网络的数千倍，并且具有良好的泛化能力[1]。ELM 的核心思想是随机生成网络的输入权值和隐层阈值，并根据最小二乘法确定输出权值。但由于其隐层参数（输入权值和隐层阈值）的随机性，难以保证得到好的结果 [2,3]。为找到最优网络参数， Zhu 等人提出了进化极端学习（evolutionary ELM，E-ELM），其中输入权重和隐层阈值通过差分方法进行优化，输出权重使用 Moore-Penrose（M-P）广义逆计算，该算法学习速度快，但推广能力较差 [4]。Huang 等人提出一种增量极端学习机（incremental ELM，I-ELM），有效的提高了网络学习速度，但测试精度稍差[5,6]。Rong 等人提出剪枝极端学习机（pruning ELM，P-ELM），该方法能够得到较好的测试精度以及简单的网络结构，但学习的时间相对较长 [7]。Emilio 等人提出的贝叶斯极端学习机（Bayesian ELM，BELM），提高了网络推广能力，但随机参数问题仍有待改进[8]。Han 等人采用混合学习算法来克服极端学习机的缺陷，利用改进后的粒子群极端学习机（Particle Swarm Optimization，PSO-ELM）来选择输入权值和隐层阈值，有较好的网络泛化能力，但模型结构复杂[9]。综合来看，针对 ELM 的讨论主要集中在两点：如何确定输入层与隐层之间的连接权以及如何确定隐层节点数。

本文主要针对 ELM 随机选取输入层和隐含层之间的连接权（输入权值）及隐层阈值的问题，提出一种新的单隐层神经网络权值确定算法——输出值反向分配算法，利用优化方法确定输入权值。经过数值实验，得出输出值反向分配算法可以解决输入参数随机设定的问题，能够得到较好的推广能力。

1　基本理论

1.1　单隐层前馈神经网络

单隐层前馈神经网络由输入层，单隐含层和输出层构成[10]。相邻层的节点由连接权重进行全连接，单隐层前馈神经网络能够表示成如下函数：

$$f(X)=\sum_{i=1}^{L}\boldsymbol{v}_i\varphi\left(\sum_{j=1}^{n}\boldsymbol{w}_{ij}\boldsymbol{x}_j+b_i\right)+v_0=\sum_{i=1}^{L}\boldsymbol{v}_i\varphi\left(\boldsymbol{W}_i^T\cdot\boldsymbol{X}+b_i\right)+v_0 \tag{1}$$

收稿日期：2018-07-27；**修回日期：**2018-09-12　**基金项目：**沪江基金资助项目（A14006）；国家级项目培养基金资助项目（16HJPY-MS02）

作者简介：刘金澎（1993-），女，吉林四平人，硕士研究生，主要研究方向为优化算法、数据分析与数据挖掘（ljpshlgdx@163.com）；田大钢（1958-），男，教授，主要研究方向为优化算法、计算智能、决策支持和决策支持系统、数据分析与数据挖掘.

其中：X=（x_1，x_2，x_3，…，x_n）^{T}表示网络的 n 维输入向量，W_i=（w_{i1}，w_{i2}，w_{i3}，…，w_{in}）^{T}表示输入层与隐层第 i 个节点的连接权（即输入权值），它的大小代表着各节点之间连接的强弱，b_i是隐层第 i 个节点的阈值（即隐层阈值），v_i表示隐层第 i 个节点到输出层的连接权（即输出权值），φ(·)表示隐层活化函数，本文使用 *Sigmoid* 函数，$W_i \cdot X$表示W_i和X的内积，$f(X)$为网络输出。输出神经元的活化函数为线性函数，所以网络的输出权向量 V=（v_1，v_2，…，v_L）^{T}可以根据最小二乘法求解，极端学习机就是这样求解的。但 W 和 b 却不易求得，文献[6][7][8]都涉及到这个问题，本文将给出一种新的算法来确定隐层参数（W、b）。

1.2 极端学习机算法

设有任意的 N 个样本（X_i，Y），i=1，2，…，N，X_i=（x_{i1}，x_{i2}，…，x_{in}）^{T} 为网络的 n 维输入向量，Y=（y_1，y_2，…，y_N）^{T}为 N 个样本输出。则样本学习问题可以表达为：

$$f(X_k) = \sum_{i=1}^{L} v_i \varphi(W_i^T \cdot X_k + b_i) + v_0 \approx y_k, k = 1,2,\cdots,N \tag{2}$$

其中：选取合适的输出权重 v_i、输入权重 W_i 以及隐层阈值 b_i，使得样本输出 y_k 与网络输出之间的误差尽量小。

因为网络输出函数为线性函数，所以式（2）就是关于 V=（v_1，v_2，…，v_L）^{T} 的线性方程组。给定输入权值和隐层阈值后，输出权重可以通过求解线性方程组求得。

上述方程组可以简写为

$$HV=Y \tag{3}$$

H 记为

$$H = \begin{bmatrix} \varphi(W_1^T \cdot X_1 + b_1) & \cdots & \varphi(W_L^T \cdot X_1 + b_L) & 1 \\ \varphi(W_1^T \cdot X_2 + b_1) & & \varphi(W_L^T \cdot X_2 + b_L) & 1 \\ \vdots & \ddots & \vdots & \\ \varphi(W_1^T \cdot X_N + b_1) & \cdots & \varphi(W_L^T \cdot X_N + b_L) & 1 \end{bmatrix}_{N\times(L+1)}$$

H 是式（2）的系数矩阵，V=（v_1，v_2，…，v_L）^{T}是输出权值向量，Y=（y_1，y_2，…，y_N）^{T}是样本的一维输出。容易知道，对几乎所有的 W，b，H 为列满秩矩阵。

特别地，当 L=N-1，即隐层节点数等于样本个数时，只要隐层输出矩阵 H 为非奇异矩阵，存在唯一的 V，满足方程组（3），因此，含有 L（L=N-1）个隐层节点的单隐层前馈神经网络能够以零误差学会 N 个样本[2,11]，也就是网络输出与样本输出完全一致。但事实上样本数一般很大，若隐层节点数等于样本数，除了导致计算量很大之外，更主要地是会出现过拟合现象，网络的泛化能力降低[1]。

当 L<N-1，即隐层节点小于样本数时，线性方程组（3）未知量个数少于方程个数，可以利用最小二乘法求解出向量 V。上述式（3）的最小二乘解为

$$\hat{V}=H^+Y=(H^TH)^{-1}H^TY$$

其中 H^+=（H^TH）$^{-1}H^T$ 为 H 的 *M-P* 广义逆矩阵[12]。

极端学习机的基本步骤如下：

a）随机选择隐层参数（网络输入权值 W_i 和隐层阈值 b_i）；

b）根据隐层参数计算隐层输出矩阵 H；

c）根据求解得到输出权值 V。

由上可见，极端学习机的一个主要难点就是解决 W，b 的选择问题。

2 基于 ELM 的输出值反向分配算法

取$v_0 = 0$。设有任意的 N 个样本（X_j，Y_j），j=1，2，…，N，X_j=（x_{j1}，x_{j2}，…，x_{jn}）^{T}为网络的 n 维输入向量，Y=（y_1，y_2，…，y_N）^{T}为 N 个样本输出。设隐层节点数为 L，隐层活化函数为 *Sigmoid* 函数，V=（v_1，v_2，…，v_L）^{T}为输出权值，W_i表示输入向量与隐层第 i 个节点的连接权（即输入权值），b_i是隐层第 i 个节点的阈值（即隐层阈值）。

取分配系数向量$\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_L)^T$，0≦$\alpha_i$≦1，i=1，2，…，L，令隐层节点 i 的输出为$y\alpha_i$，即

$$\varphi(W_i^T \cdot X + b_i) = y\alpha_i \tag{4}$$

则

$$W_i^T \cdot X + b_i = \varphi^{-1}(y\alpha_i) \tag{5}$$

本文选取的活化函数为 *Sigmoid* 函数，该函数值域在（0，1），为使$\varphi^{-1}(y\alpha_i)$有定义，需要对样输出 y 进行转换，使其值在（0，1）之间。

将样本代入式（4），可以得到如下方程组：

$$\begin{cases} \varphi(W_i^T \cdot X_1 + b_i) = y_1\alpha_i \\ \varphi(W_i^T \cdot X_2 + b_i) = y_2\alpha_i \\ \vdots \\ \varphi(W_i^T \cdot X_N + b_i) = y_N\alpha_i \end{cases}$$

再根据式（5）得到如下方程组：

$$\begin{cases} W_i^T \cdot X_1 + b_i = \varphi^{-1}(y_1\alpha_i) \\ W_i^T \cdot X_2 + b_i = \varphi^{-1}(y_2\alpha_i) \\ \vdots \\ W_i^T \cdot X_N + b_i = \varphi^{-1}(y_N\alpha_i) \end{cases} \tag{6}$$

当α给定时，式（6）是关于 W，b 的线性方程组，所以可以通过最小二乘法确定输入权值 W_i 和隐层阈值 b_i。

为表述简洁，记上述方程组为

$$\widetilde{W}_i = \binom{W_i}{b_i}, \widetilde{X}_j = \binom{X_j}{1}$$

这里$\widetilde{W}_i$是 n+1 维向量，i=1，2，…，L，$\widetilde{X}_j$是 n+1 维向量，j=1，2，…，N。则

$$\widetilde{W} = \begin{pmatrix} W_1 & W_2 & \cdots & W_L \\ b_1 & b_2 & \cdots & b_L \end{pmatrix} = (\widetilde{W}_1 \quad \widetilde{W}_2 \quad \cdots \quad \widetilde{W}_L)$$

$$\widetilde{X} = \begin{pmatrix} X_1 & X_2 & \cdots & X_N \\ 1 & 1 & \cdots & 1 \end{pmatrix} = (\widetilde{X}_1 \quad \widetilde{X}_2 \quad \cdots \quad \widetilde{X}_N)$$

其中：$\widetilde{W}$是 n+1 行 L 列矩阵，$\widetilde{X}$是 n+1 行 N 列矩阵。网络隐层神经元输出矩阵 H 可记为

$$H=\begin{bmatrix}\varphi(\widetilde{W}_1^{\mathrm T}\cdot\widetilde{X}_1) & \cdots & \varphi(\widetilde{W}_L^{\mathrm T}\cdot\widetilde{X}_1)\\ \varphi(\widetilde{W}_1^{\mathrm T}\cdot\widetilde{X}_2) & & \varphi(\widetilde{W}_L^{\mathrm T}\cdot\widetilde{X}_2)\\ \vdots & \ddots & \vdots\\ \varphi(\widetilde{W}_1^{\mathrm T}\cdot\widetilde{X}_N) & \cdots & \varphi(\widetilde{W}_L^{\mathrm T}\cdot\widetilde{X}_N)\end{bmatrix}_{N\times L}=\varphi(\widetilde{X}^{\mathrm T}\widetilde{W})$$

系数矩阵中，当$A=(a_{ij})_{m\times n}$时，用$\varphi(A)$表示矩阵$(\varphi(a_{ij}))_{m\times n}$。

由此，方程式（6）可以简写为

$$\widetilde{X}^{\mathrm T}\widetilde{W}_i=\varphi^{-1}(Y\alpha_i) \tag{7}$$

其中：$\varphi^{-1}(Y\alpha_i)=(\varphi^{-1}(y_1\alpha_i),\varphi^{-1}(y_2\alpha_i),\cdots,\varphi^{-1}(y_N\alpha_i))^{\mathrm T}$。因此（7）式中的最小二乘解为

$$\widetilde{W}_i=(\widetilde{X}^{\mathrm T})^{+}\varphi^{-1}(Y\alpha_i)=(\widetilde{X}\widetilde{X}^{\mathrm T})^{-1}\widetilde{X}\varphi^{-1}(Y\alpha_i)$$

$$i=1,2,\cdots,L \tag{8}$$

式（8）是含有输出值分配系数向量α的表达式。确定α，使其在满足约束条件情况下，最小化下列目标函数：

$$min\frac{1}{N}\left\|\sum_{i=1}^{L}v_i\varphi(\widetilde{X}^{\mathrm T}(\widetilde{X}^{\mathrm T})^{+}\varphi^{-1}(Y\alpha_i))-Y\right\|^2$$

这里取L_2范数。

这样，求解输入权值和隐层阈值的问题，就转换为求解含有输出值分配系数向量α的优化问题：

$$\begin{cases}min\frac{1}{N}\left\|\sum_{i=1}^{L}v_i\varphi(\widetilde{X}^{\mathrm T}(\widetilde{X}^{\mathrm T})^{+}\varphi^{-1}(Y\alpha_i))-Y\right\|^2\\ 0\le\alpha_i\le 1\end{cases} \tag{9}$$

由式（9）可以求出最优的α。确定α以后，根据式（8）计算$\widetilde{W}_i$，再根据极端学习机算法求出V：

$$\hat{V}=H^{+}Y=(H^{T}H)^{-1}H^{T}Y \tag{10}$$

令$e(V,\alpha)=\frac{1}{N}\left\|\sum_{i=1}^{L}v_i\varphi(\widetilde{X}^{\mathrm T}(\widetilde{X}^{\mathrm T})^{+}\varphi^{-1}(Y\alpha_i))-Y\right\|^2$，输出值反向分配算法的基本步骤如下：

输入：X、Y、L、ε；

输出：W、V；

a）随机选择输出权值$V=(v_1^0, v_2^0, \cdots, v_L^0)^{\mathrm T}$；

b）根据式（9）求出最优输出值分配系数向量$\alpha=(\alpha_1^0,\alpha_2^0,\cdots,\alpha_L^0)^{\mathrm T}$；

c）根据式（8）计算输入权值和隐层阈值$\widetilde{W}_i$；

d）由式（10）确定新的输出权值V。

e）若$e(V,\alpha)<\varepsilon$，则算法终止，否则返回步骤b）。

容易知道，以上算法显然是收敛的。事实上，$e(V^0,\alpha^0)\ge e(V^1,\alpha^0)\ge e(V^1,\alpha^1)\ge e(V^2,\alpha^1)\ge\cdots\ge 0$。

本文提出的输出值反向分配算法由于V是由最小二乘求得，保留了极端学习机的优点，同时W是通过α用$\widetilde{W}_i=(\widetilde{X}^{\mathrm T})^{+}\varphi^{-1}(Y\alpha_i)=(\widetilde{X}\widetilde{X}^{\mathrm T})^{-1}\widetilde{X}\varphi^{-1}(Y\alpha_i)$算得，$\alpha$由优化算法求得，本文算法可望能够获得比随机选择好的结果。同时，在算法的计算结果中，如果某一个$\alpha_i=0$，则相应的隐层节点可以删除，这使得该算法有选择隐层节点数的作用。

式（9）并不能保证是凸优化问题，但计算效果比较好，尤其对分类问题，其中的原因有待进一步研究。

3 数值实验

3.1 实验数据

本文选取的实验数据均来源于UCI机器学习数据库[13]，该数据库中的数据均为实际问题的真实数据，并多被用于机器学习研究。本文选取了常用的Iris、Wine、Pima Indians Diabetes和Wisconsin Breast Cancer数据集、Heart、Balance Scale、Haberman's Survival和User Knowledge Modeling数据集，其中测试样本均是随机选取所得，如表1所示。

表1 数据集基本属性

数据集名称	训练样本	测试样本	属性	分类
Iris	100	50	4	3
Wine	118	60	13	3
Pima I.D	538	230	8	2
Wisconsin B.C	379	190	30	2
Heart	180	90	13	2
Balance Scale	425	200	4	3
Haberman's S	204	102	3	2
User K.M	258	145	5	4

3.2 实验过程及结果

a）Iris数据集。该算例将样本分为三类，分别对应1、2、3。输出值反向分配算法中活化函数为*Sigmoid*函数，因此需要将样本数据的期望输出调整至（0，1），根据下式进行转换：

$$y_0=\frac{y-min(y)}{max(y)-min(y)} \tag{11}$$

y_0为变换后的期望输出，y为原本样本期望输出。

应用本文输出值反向分配算法进行实验，构造含有三个隐层节点的单隐层网络，即式（2）中$L=3$，随机给定输出权值，将训练样本以及测试样本代入算法，根据上面所述的学习算法过程，求得最终网络输入权值

$$W=\begin{pmatrix}-0.0780 & 0.0824 & 0.0824\\ -0.0515 & 2.0980 & 2.0982\\ 0.1320 & -0.1784 & -0.1784\\ 0.3117 & 10.2198 & 10.2203\end{pmatrix}^{\mathrm T}$$

隐层阈值$b=(-27.4835\ -12.9238\ -12.9245)^{\mathrm T}$，网络最终输出权值$V=(1.1625\times10^{12}\ 2.2334\times10^{3}\ -2.2333\times10^{3})^{\mathrm T}$。

b）Wine数据集。该算例将样本分为3类，实验前，将类别进行如式（11）的转换，首先构造一个含有20个隐层节点的单隐层网络，根据本文的输出值反向分配算法进行学习，最后α_1，α_4，α_9，α_{10}，α_{13}，α_{14}，α_{15}，α_{16}，α_{17}，α_{18}均为零，删除对应的隐层节点，得到含有10个隐层节点的单隐层网络。网络输入权值为

$$W=\begin{pmatrix} -0.0094 & -0.0029 & -0.0028 & -0.0028 & -0.0016 & -0.0033 & -0.0033 & -0.0028 & -0.0028 & -0.0033 \\ 8.4085\times10^{-5} & 5.6784\times10^{-4} & 4.3521\times10^{-4} & 4.9985\times10^{-4} & 0.0045 & 0.0014 & 0.0014 & 4.3188\times10^{-4} & 4.6795\times10^{-4} & 0.0013 \\ -0.0269 & -0.0078 & -0.0076 & -0.0076 & -0.0014 & -0.0085 & -0.0084 & -0.0076 & -0.0075 & -0.0085 \\ 0.0031 & 0.0011 & 0.0010 & 0.0011 & 0.0022 & 0.0016 & 0.0016 & 0.0010 & 0.0011 & 0.0015 \\ 1.4270\times10^{-5} & -5.9930\times10^{-6} & -3.6486\times10^{-6} & -4.8417\times10^{-6} & -8.3720\times10^{-5} & -2.0872\times10^{-5} & -2.2064\times10^{-5} & -3.5667\times10^{-6} & -4.2894\times10^{-6} & -1.8322\times10^{-5} \\ 0.0156 & 0.0067 & 0.0060 & 0.0063 & 0.0185 & 0.0102 & 0.0105 & 0.0060 & 0.0061 & 0.0097 \\ -0.0243 & -0.0117 & -0.0104 & -0.0109 & -0.0399 & -0.0193 & -0.0197 & -0.0104 & -0.0106 & -0.0182 \\ -0.0178 & -0.0100 & -0.0087 & -0.0092 & -0.0408 & -0.0176 & -0.0181 & -0.0086 & -0.0089 & -0.0164 \\ 0.0096 & 0.0039 & 0.0036 & 0.0037 & 0.0101 & 0.0059 & 0.0060 & 0.0036 & 0.0036 & 0.0056 \\ 0.0022 & 0.0015 & 0.0012 & 0.0013 & 0.0070 & 0.0027 & 0.0028 & 0.0012 & 0.0013 & 0.0025 \\ -2.7572\times10^{-4} & -0.0024 & -0.0019 & -0.0021 & -0.0195 & -0.0059 & -0.0062 & -0.0018 & -0.0020 & -0.0054 \\ -0.0145 & -0.0068 & -0.0061 & -0.0064 & -0.0223 & -0.0110 & -0.0113 & -0.0061 & -0.0062 & -0.0104 \\ -5.1577\times10^{-5} & -1.9292\times10^{-5} & -1.7822\times10^{-5} & -1.8376\times10^{-5} & -3.8219\times10^{-5} & -2.6942\times10^{-5} & -2.7344\times10^{-5} & -1.7859\times10^{-5} & -1.7985\times10^{-5} & -2.6016\times10^{-5} \end{pmatrix}^{T}$$

隐层阈值 b=（-0.0390 0.0102 -0.0471 -0.0094 0.1210 0.0812 0.0835 -0.0508 -0.0234 0.0754）^{T}，网络的输出权值 V=（7.0693×10^{5} -1.9033×10^{10} 8.0851×10^{10} 6.6569×10^{10} -8.4713×10^{5} -2.1497×10^{10} 1.2798×10^{10} -5.6410×10^{10}）^{T}。

c）Pima Indians Diabetes 数据集。该算例的样本期望输出为 0、1 两类，首先构造一个含有 10 个隐层节点的单隐层网络，利用本文中的网络算法进行学习，得到α_3，α_4，α_5都为零，所以删除对应的隐层节点，得到含有 7 个隐层节点的单隐层网络。网络输入权值为

$$W=\begin{pmatrix} 0.1450 & 0.1455 & 0.1467 & 0.4055 & 0.1369 & 0.1389 & 0.1341 \\ 0.0352 & 0.0353 & 0.0356 & 0.0984 & 0.0332 & 0.0337 & 0.0325 \\ -0.0133 & -0.0134 & -0.0135 & -0.0373 & -0.0126 & -0.0128 & -0.0123 \\ 2.9216\times10^{-4} & 2.9328\times10^{-4} & 2.9559\times10^{-4} & 8.1720\times10^{-4} & 2.7582\times10^{-4} & 2.7995\times10^{-4} & 2.7030\times10^{-4} \\ -0.0011 & -0.0011 & -0.0012 & -0.0032 & -0.0011 & -0.0011 & -0.0011 \\ 0.0914 & 0.0918 & 0.0925 & 0.2558 & 0.0863 & 0.0876 & 0.0846 \\ 0.9595 & 0.9632 & 0.9708 & 2.6839 & 0.9095 & 0.9194 & 0.8877 \\ 0.0059 & 0.0060 & 0.0060 & 0.0166 & 0.0056 & 0.0057 & 0.0055 \end{pmatrix}^{T}$$

隐层阈值 b=(-16. 5627 -16. 5821 -16. 6218 -25. 6151 -16. 2811 -16. 3522 -16. 1858)^{T}，网络的输出权值 V=(-4.9056×10^{8} 5.8249×10^{8} -1.2476×10^{8} -0. 1772 -7.7607×10^{7} 9.5411×10^{7} 1.5032×10^{7})^{T}。

d）Wisconsin Breast Cancer 数据集。该数据集样本分为良性和恶性两类，分别对应 1 和 0，构造含有 1 个隐层节点的单隐层网络，得到的网络输入权值为 W=（-1.4687×10^{-10} 6.5666 0.1334 -0.5236 -0.0174 -3.8815 63.0322 -7.8507 -31.6170 -26.3393 18.1990 -1.0747 0.4935 0.8863 -0.0440 -399.0794 26.5462 59.0644 -122.0129 -204.8474 107.1777 -5.5983 -0.2940 0.0673 0.0297 19.8680 1.6532 -9.6924 -36.9432 21.4134 -107.7062 ），隐层阈值 b=40.6548，网络输出权值 V=1.0269。

e）Heart 数据集。该数据集含有 13 个属性值，即为样本输入，样本输出为样本所属类别，该样本数据分为 1、2 两类。经过上述式（11）的转换以后，构造含有 2 个隐层节点的单隐层网络，得到的网络输入权值为

$$W=\begin{pmatrix} -0.0041 & -0.0709 \\ 0.1570 & 2.7261 \\ 0.0909 & 1.5789 \\ 0.0017 & 0.0291 \\ 4.6387\times10^{-4} & 0.0081 \\ 0.0151 & 0.2628 \\ 0.0138 & 0.2388 \\ -0.0022 & -0.0389 \\ 0.0594 & 1.0317 \\ 0.0430 & 0.7474 \\ 0.0272 & 0.4717 \\ 0.1630 & 2.8296 \\ 0.0426 & 0.7403 \end{pmatrix}^{T}$$

隐层阈值 b=(-24.6254 -3.4921)^{T}，输出权值 V=（2.5763×10^{10} 0.4878)^{T}。

f）Balance Scale、Haberman's Survival 和 User Knowledge Modeling 数据集。Balance Scale 数据集为多元分类，样本期望输出分别为-1、0 和 1；Haberman's Survival 数据集为二元分类，样本期望输出分别对应 1 和 2；User Knowledge Modeling 数据集为多元分类，样本期望输出分别对应 1、2、3 和 4，在均进行如上式（11）的转换之后，分别构造含有 4、2 和 7 个隐层节点的单隐层网络进行学习。实验结果见表 2。限于篇幅，略去了网络输入权值、隐层阈值和输出权值。

将网络输出值 y 进行四舍五入后得到 y^*，并与网络期望输出值进行比较，得到如下实验结果，如表 2 所示。

表 2　数据集实验结果

数据集	RMSE	TestRMSE	训练精度	测试精度	节点数
Iris	0.2360	0.1975	95%	98%	3
Wine	0.1285	0.2035	100%	98.33%	10
Pima I.D	0.3974	0.3735	77.32%	79.57%	7
WisconsinB.C	0.1284	0.1850	97.89%	95.79%	1
Heart	0.3524	0.3787	85%	86.67%	2
Balance Scale	0.3585	0.3357	90.59%	94%	4
Haberman's S	0.4257	0.4408	73.04%	78.43%	2
User K.M	0.2349	0.2137	94.19%	98.62%	7

根据本文提出的算法对于以上 8 个算例的学习情况，在保证较低的训练误差以及测试误差情况下，能够得到较高的分类正确率。此外，在这些例子中，网络最多含有 10 个隐层节点，最少含有 1 个隐层节点，在计算过程中，网络取多个隐层节点进行实验，会出现某些$\boldsymbol{\alpha}_k=0$的情况，这就意味着其对应的隐层节点可以删除。

以上实验数据皆为分类数据集，主要因为本文的算法为输出值反向分配算法。按照本文的思路，当样本是分类问题时，输出值只有少数几个数字，比较容易用几个$\boldsymbol{\alpha}_k$实现分配。所以算法对分类问题的效果比较好。而对于样本输出为连续值的回归数据集，拟合的效果比较差。

针对文中 Iris、Wine、Pima Indians Diabetes 和 Wisconsin Breast Cancer 算例，将本文所提出的算法与其他文献中提出的多种改进的极端学习机算法：文献[14]中优化剪枝极端学习机（OP-ELM），文献[15]中分层极端学习机（H-ELM）和传统极端学习机（ELM）、文献[16]中稀疏贝叶斯极端学习机（SBELM）

以及文献[17]中基于粒子群优化算法极端学习机（PSO-ELM）进行实验对比，如表 3 所示。

表 3　分类实验结果比较

数据集	算法	测试精度	节点数
Iris	本文	98%	**3**
	OPELM	95%	4~17
	HELM	**100%**	N1=N2=20; N3=200
	ELM	92%	200
	SBELM	98%	2.4×3
Wine	本文	98.33%	10
	OPELM	90.7%	不详
	HELM	**100%**	N1=N2=20; N3=500
	ELM	95%	500
	SBELM	99.41%	**3.3×3**
Pima I.D	本文	79.57%	**7**
	OPELM	74.9%	不详
	HELM	**80.47%**	N1=N2=10; N3=200
	ELM	76.95%	200
	SBELM	78.66%	8
	PSO-ELM	76.38%	10
Wisconsin B. C	本文	95.79%	**1**
	OPELM	95.6%	不详
	SBELM	**97.22%**	5.6

对比各个实验结果可以看出，本文算法与其他上述算法的精度大致相同。除了 Wine 数据集中，本文算法与 SBELM 算法的隐层节点数略有差异以外，对于其它数据集，本文算法所需的隐层节点数最少。根据统计学习理论，节点数少的网络推广能力更好的概率更大。

4　结束语

本文提出了一种输出值反向分配算法来求得单隐层神经网络的输入权值与隐层阈值。输入权值通过分配系数向量$\boldsymbol{\alpha}$确定，$\boldsymbol{\alpha}$通过优化方法求得。实验结果表明，本文算法在保证有较好的学习质量情况下，可以获得结构比较简单的单隐层网络。

参考文献：

[1] Huang GuangBin, Zhu QinYu, Siew C K. Extreme learning machine: theory and applications [J]. Neurocomputing, 2006, 70 (1–3): 489-501.

[2] 牛培峰，马云鹏，刘魏岩，等. 极端学习机算法的改进及应用研究 [J]. 燕山大学学报, 2015 (2): 127-132. (Niu Peifeng, Ma Yunpeng, Liu Weiyan, *et al.* Improvement and application of extreme learning machine algorithm [J]. Journal of Yanshan University, 2015 (2): 127-132.)

[3] 卢海峰，卫伟，杨梦月. 局部感知的类限制极限学习机 [J/OL]. 计算机应用研究, 2018, 35 (10) . http://www. arocmag. com/article/02-2018-10-009. html. (Lu Haifeng, Wei Wei, Yang Mengyue. Receptive field class-constrained extreme learning machine. [J/OL]. Application Research of Computers, 2018, 35 (10) . http://www. arocmag. com/article/02-2018-10-009. html.)

[4] Zhu Qinyu, Qin A K, Suganthan P N, *et al.* Evolutionary extreme learning machine [J]. Pattern Recognition, 2005, 38 (10): 1759-1763.

[5] Huang Guangbin, Chen Lei, SiewC. K. Universal approximation using incremental constructive feedforward networks with random hidden nodes [J]. IEEE Trans on Neural Networks, 2006, 17 (4): 879-892.

[6] Huang Guangbin, Chen Lei. Convex incremental extreme learning machine [J]. Neurocomputing, 2007, 70 (16-18): 3056-3062.

[7] Rong H J, Ong Y S, Tan A H, *et al.* A fast pruned-extreme learning machine for classification problem [J]. Neurocomputing, 2008, 72: 359-366.

[8] Soria Olivas E, Gomez Sanchis J, Martin J D, *et al.* BELM: Bayesian extreme learning machine [J]. IEEE Trans on Neural Networks, 2011, 22 (3): 505-509.

[9] Han F, Yao H F, Ling Q H. An improved evolutionary extreme learning machine based on particle swarm optimization [J]. Neurocomputing, 2013, 116: 87-93.

[10] Hagan M T, Demuth H B. 神经网络设计 [M]. 北京：机械工业出版社, 2002: 8-15.

[11] Huang Gao, Huang Guangbin, Song Shiji, *et al.* Tends in extreme learning machines [J]. Neural Networks, 2015, 61 (C): 32-48.

[12] SerreD, Matries: Theory and Applications [M]. Spring, New York, 2002: 18-32.

[13] UCI Machine Learning Repository [DB/OL]. http://archive. ics. uci. edu/ml/datasets. html.

[14] Miche Y, Sorjamaa A, Bas P, *et al.* OP-ELM: Optimally Pruned Extreme Learning Machine [J]. IEEE Trans on Neural Networks, 2010, 21 (1): 158-162.

[15] Tang Jiexiong, Deng Chenwei, Huang Guangbin. Extreme Learning Machine for Multilayer Perceptron [J]. IEEE Trans on Neural Networks and Learning Systems, 2016, 27 (4): 809-821.

[16] Luo Jiahua, Vong C M, Wong P K. Sparse Bayesian Extreme Learning Machine for Multi-classification [J]. IEEE Trans on Neural Networks and Learning Systems, 2014, 25 (4): 836-843

[17] 顾秀君. 基于粒子群优化的极端学习机的研究及其应用 [D]. 镇江：江苏大学, 2013. (Gu Xiujun. A Study of Extreme Learning Machine Based on Particle Swarm Optimization And Its Application [D]. Zhenjiang: Jiangsu University, 2013)